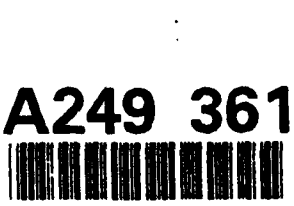


Analysis of Online Algorithms for Organ Allocation

by

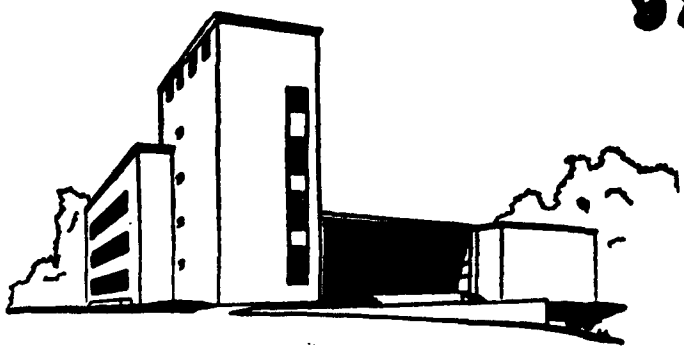
Shmuel Ur*

Carnegie Mellon University

PITTSBURGH, PENNSYLVANIA 15213

GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION

WILLIAM LARIMER MELLON, FOUNDER



Management Sciences Research Report No. MSRR 579

Analysis of Online Algorithms for Organ Allocation

by

Shmuel Ur*

* Graduate School of Industrial Administration
Carnegie Mellon University
Pittsburgh, PA 15213

This report was prepared as part of the activities of the Management Science Research Group, Carnegie Mellon University, under Contract No. N00014-85-K-0198 NR 047-048 with the Office of Naval Research. Reproduction in whole or in part is permitted for any purpose of the U. S. Government.

Management Sciences Research Group
Graduate School of Industrial Administration
Carnegie Mellon University
Pittsburgh, PA 15213

Analysis of Online Algorithms for Organ Allocation

by

Shmuel Ur

Abstract

In this paper we discuss the need for a refinement to the existing organ allocation policy. Under the current organ allocation algorithm - the Point System - whenever an organ becomes available each patient scores points for that organ according to a multitude of criteria, and the patient with the highest score gets the organ.

We will show that in order to select the patient that should receive the organ we must look not only at each patient individually but also at the current collection of patients. Current work in progress shows that the expected distribution of patients as well as the expected distribution of organs should affect the allocation policy.

We suggest an objective function for a case when there are no overwhelming considerations to prefer one patient over the another (i.e they have the same score, or are very close). The objective function we analyze is the number of organs utilized. The reason is that organs are considered to be a scarce resource not to be wasted. We will select a patient in such a way that as few possible organs will be wasted in the future.

In this paper we will solve, using the online model, some simplifications of the general problem. We will explain the intuition that we glean from them. We will also present some preliminary results from analyzing the problem using statistical and simulation methods.

1. Introduction

The models to be discussed in this paper are simplifications of the organ allocation problem. Simply stated the organ allocation problem is: decide which patient should receive each organ. This is an online problem since an organ must be used almost immediately else it is wasted.

Under the current algorithm used today for organ allocation [6] - the Point System - whenever an organ becomes available each patient scores points for that organ according to a multitude of criteria, and the one with the highest score gets the organ .

To quote from a UNOS (United Network for Organ Sharing) flier [7] - "Who gets to be number 1 on the waiting list rather than 1,000? Factors affecting a patient's placement on the waiting list include medical urgency, length of time on the waiting list, distance between the patient and the transplant center and the donor and the transplant center, organ weight and tissue matching." Every patient scores some points using this system and the patient with most points will get the organ.

One can raise a question is this allocation policy optimal? What do we actually mean when we say optimal in this context?

In order to be able to compare different algorithms we need an objective function to optimize. This is a major pitfall. To our knowledge this problem has not previously been analyzed because of the lack of an acceptable objective function [1]. Legally the objective function must be one that does not discriminate on grounds of sex and race. Moral as well as legal considerations require that people who are in immediate need will get the organ first . All these considerations make it hard to agree on a global objective function. Our perpose is to refine the existing system and suggest an objective function for a case when there are no overwhelming reasons to prefer one patient over the rest (i.e. when the current system will give maximum score to more then one patient) . The objective function we chose to analyze is the number of organs utilized. The reason is that organs are considered to be a scarce resource not to be wasted. We will select the patient in such a way that as few possible organs will be wasted in the future. We will also refer to the case of kidney transplants where the option of cleaning the blood by circulating it through a hemodialysis (artificial kidney) machine can be considered. We will use the criterion that as few machines as possible should be used while letting as many patients as possible enjoy a transplanted organ. In this special case this criterion makes sense form monetary as well as humanitarian considerations.

We will show that in order to select the patient that should receive the organ we must look not only at each patient individually but also at the current and expected distribution of patients as well as the expected distribution of organs.

We will usually refer to patients by their blood type. It is therefore necessary to know the possible ways in which an organ can be used. Figure 1 shows the possible donations according to blood type.

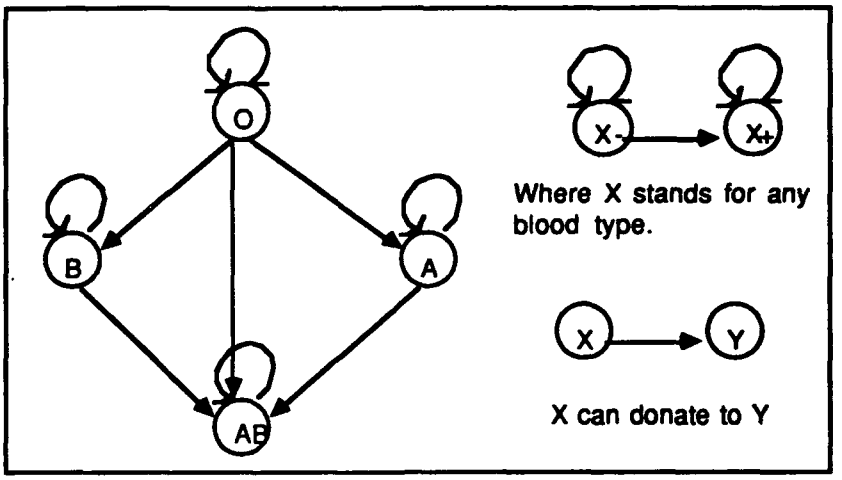


Figure 1. Possible donations according to blood type

The following two examples show that the expected distribution of organs should be considered. In the first, two patients are waiting for an organ, one of blood type **O-** and the other of blood type **O+**. An organ becomes available of blood type **O-**. Assume that both patients can benefit equally from receiving this organ. It makes sense to give it to the patient with blood type **O-** since his chances of finding a match in the future are much smaller (the percent of people with blood type **O-** is very small).In the second, two patients, one of blood type **A** and one of blood type **B**, are waiting for an organ. Let's assume that the patient of blood type **A** has priority. Assume that an organ of blood type **O** becomes available and immediately afterwards an organ of blood type **A** becomes available. According to the current system the patient with blood type **A** will get the organ with type **O** and the other organ will be wasted (unless somebody is watching to eliminate such mistakes). This is to the disadvantage of both patients: the patient with blood type **B** will not receive an organ at all, while the patient with blood type **A** will receive an **O** organ, which is slightly less suitable for him than an **A** organ. If the organ of blood type **A** is not immediately available but statistically is expected to become available soon then it still seems reasonable to give the organ of blood type **O** to the patient of blood type **B**. This will not be done if the future distribution of organs is not considered. Similar reasoning can show that the expected distribution of patients should be consedered.

The following examples shows that the current distribution of patients should be considered. Let there be ten patients with blood type **A** and a single patient of blood type **B** all of which are in a non-critical condition. An organ of type **O** apears. Common sense tells us that one of the patients with blood type **A** should get the organ. The reason is that otherwise a future organ of type **B** might be wasted. A similar example of a more general kind is a disaster in some area. More resources are sent to this area because of the large number of patients. Each patient by himself may not be in critical condition, but since it is known that the resources of this area are not sufficient to handle the emergency, resources are sent to the area so that help can be given to

patients in non-critical conditions, to avoid worsening their condition. The usual assumption - that the rate of treatment is sufficient to ensure that patients in non-critical conditions do not reach critical stages - is invalid in this case. Therefore a patient with a non-critical condition in this area is given priority over a similar patient from a different area.

We will analyze the problem as an online problem and prove that the current distribution of patients is an important consideration. We will remark on some results we obtained, using stochastic methods as well as simulations, that show that the expected distribution of patients and the expected distribution of organs are also important considerations.

We will present the theoretical solutions to different models of the problem, and will present the intuitions behind the solutions and how they apply to the real problem.

We will show that no algorithm that optimizes locally i.e. the score for each patient is independent of the other patients (the point system is one), can be optimal for both the problem of maximizing the number of organs, and minimizing the number of machines.

By looking at simulation models [3] it is evident that patient distribution has an impact on the problem. It was found out in a simulation of waiting lines for kidney transplants that "the number of hard to match cases is not reduced in direct proportion to increases in the number of donated kidneys". We will show how this problem can be addressed by modifying the allocation policy.

We realize that for most organs the recipient will be selected by legal or ethical considerations , or there will be an overwhelming medical reason to prefer a specific patient. However, for the cases in which the current system offers no clear cut choice, we will offer a system which will result in a higher percentage of the organs being used.

We start by presenting a general Queue model that has most of the properties of the original problem (Section 2). The main difference between this model and organ allocating is that we make the simplification that an organ must fit the patient perfectly or it will not be used. This restriction on the model becomes more and more appropriate with the use of Azathioprine and other new treatments [9]. If there is a fit between the patient and the organs these new drugs make the probability of rejection small (we do not have to consider donations from relatives as they decide on the recipient). We will show a solution to simplified versions of this model, with no time limit on the patients corresponding to the examples in the introduction (Sections 3 and 4) and will present the intuition of why the ideas of this solution carry to the general problem. We will then present a situation in which patients must be served within a given number of turns (Section 5). We will show how the competitive factor of any algorithm that can be represented by a finite automata can be found in finite time (Section 6).

2. The queue model

The queue model is the general online model corresponding to the organ allocation problem. It is more general and can be used to describe other scenarios in which balking from a queue can occur.

In this model servers correspond to organs and customers correspond to patients.

We have **N** queues and **K** types of servers. Each server can serve in a subset of the queues. A customer arrives to a specific queue and has an attached value and a time; if the customer is not served within that time (in turns) he drops from the queue. Each turn a variable number of customers arrive as well as a variable number of servers. Each server that is not used is discarded.

The objective function is to maximize the total expected value of customers served each turn. If all values are the same this is equivalent to maximizing the number of servers used.

An example of a case in which this model can be applied can be found in [4] where kidneys are matched by blood types in the same way that blood is matched between donor and recipient.

Theorem 2.1: In any algorithm in any turn there is no advantage to not using a server.

Proof: Stems from the fact that all servers serve just as well.

Greedy algorithm: Each server serves the customer with the highest value available.

Theorem 2.2: Optimizing the Greedy algorithm within a turn is polynomial.

Proof: Optimizing within a turn is equivalent to the problem of weighted bipartite matching were the vertices on one side are the servers, the vertices on the other side are the customers and there exists an edge between a server and a customer if that server can serve this customer and its weight is the value of the customer. This problem is known to be polynomial .

Theorem 2.3: The Greedy algorithm has a competitive factor of two.

Proof: The best that the optimal algorithm can do is serve all the customers the Greedy algorithm served and all the customers the Greedy algorithm should have served. This comes out to at most twice the value of the customers the greedy algorithm served.

An example of this worst case happening is when every even turn one customer arrives to queue **A**, one customer arrives to queue **B** (both willing to wait one turn) and one server of type **O** (which can serve in both queues). If the algorithm chooses to serve the customer in Queue **A** then on the following odd turn an **A** type server appears else a **B** type server will appear. The optimal algorithm can serve a customer each turn while the Greedy algorithm will be able to serve only on even turns. The competitive factor of the Greedy algorithm is therefore two.

Theorem 2.4: There is no deterministic algorithm with a competitive factor better than two.

Proof: See the explanation above for the Greedy algorithm.

This result holds even if we restrict the model to having an equal value for all customers.

Theorem 2.5: There is no non-deterministic algorithm with a competitive factor better than 4/3.

Proof: On every even turn, let one customer arrive to queue **A**, one customer arrive to queue **B** (both willing to wait one turn) and one server of type **O** (which can serve in both queues). On the following odd turn a single server, with a probability of 50% to be of type **A** and a probability of 50% of being of type **B**, appear. The non-deterministic algorithm can serve a customer on the odd turn at most one out of every two turns (in probability). The competitive factor of any non-deterministic algorithm is therefore not better than 4/3.

In the real world a non perfect organ might be used. We can modify our model to reflect this fact by specifying a fit vector between the servers and the customers and by modifying the objective function to be the total value of the customers each multiplied by the fit between the customer and the server that served him.

Theorem 2.6: There is no deterministic competitive algorithm for this modified model.

Proof: Assume by way of contradiction that there exists a deterministic competitive algorithm for this modified model with a constant **c**.

Let's look at the following sequence. At time zero a customer arrives to one of the queues. The value of this customer is $(c+1)^2$, and the attached time within which it must be served is one turn. At that time a server arrives, and the fit between the server and the customer is $1/(c+1)$. If the algorithm chooses to use the server then the adversary will produce a perfect organ on the next turn. If it chooses not to use it, no organs will appear on the next turn. We can repeat this sequence every even turn and therefore there is no competitive algorithm.

Definition: A memoryless algorithm is one that cannot use any historical information in its decision rules.

Theorem 2.7: There is no memoryless non-deterministic competitive algorithm for this modified model against an oblivious adversary.

Proof: Let **R** be a non-deterministic memoryless competitive algorithm with a competitive factor of **c**. Let a customer with a value of $(2c)^2$ and an attached time $M>>c^2$ and a server with a fit between it and the customer of $1/(2c)$ arrive. Let the probability that the server will be used by **R** be p (which is known to the adversary). If $p<1/c$ then no more servers will be supplied till the M turn. The competitive factor will be **p**, contradiction. Assume $p>1/c$, the algorithm will supply servers with fit $1/(2c)$ for the next c^2 turns. Then it will supply a server with a perfect fit. The probability that none of the servers of fit $1/(2c)$ were used is

$$(1-p)^{c^2} < (1-1/2c)^{c^2} < 1/4c$$

The optimal strategy for that sequence is to wait for a server with a good fit. The competitive factor of the **R** algorithm is (value of optimal algorithm per sequence)/(expected value of **R** algorithm) which is greater than $(2c)^2/((2c)^2*(1/(4c)+1/(2c))) = 4c/3$ - in contradiction to **R** having a competitive factor of **c**. We needed to specify $M>>c^2$ in order that **R** will not be able to change his policy by using the life expectancy of the patient (which almost doesn't change while servers are being used).

Comment: Because we will be working with a 0-1 fit, and because we have multiple customers the problem of choosing whether to use an organ [2] is not discussed further in this paper.

We will now present simplified models corresponding to the different examples in the introduction. We will explain the information gained from calculating the competitive factor of different algorithms for those models.

3. A two queue two server model

In this model there are two queues **A** and **O**. There are two types of servers: **O** that can serve in both queues, and **A** that can serve only in queue **A**. Each customer has an attached value. In this scenario we will assume that these values are equal. Each new turn any number of new customers appear divided between the two queues (in any way) and any number of servers divided between **A** and **O** (in any way). These customers have no attached time (i.e., attached time equals infinity). The objective function is to maximize the average number of servers used per turn over an infinite sequence (same as value).

O algorithm: Use server **O** on a customer in queue **O** if possible, else on customers in queue **A**.

Theorem 3.1: **O** algorithm is optimal.

Proof: In this algorithm the number of customers in queue **A** is as large as possible. Therefore as few as possible servers of type **A** will be wasted. We know that servers of

type **O** are never wasted therefore the **O** algorithm is optimal. This result is counter intuitive with regard to our problem. We want the queues of customers to be as short as possible, therefore a refinement to the model is necessary.

We will refine the model by allowing each customer to have an attached time in which it must be served. Is the **O** algorithm still optimal?

Theorem 3.2: **O** algorithm has a competitive factor of two for the case where each customer has an attached time.

Proof of upper bound: Let two customers arrive; one at queue **A** that must be served immediately and one in queue **O** that can be served within one turn. At the same turn let one server of type **O** arrive. This server will serve the customer in queue **O**. On the next turn let a server of type **A** arrive. The new server will find that there are no customers to serve and therefore only one of the customers will be served. The optimal algorithm will be able to serve both of them. We can repeat this sequence and thus get a competitive factor of two.

Proof of lower bound: Whenever a server of type **A** is wasted by the **O** algorithm (servers of type **O** cannot be wasted), it means that the customer that it should have served, was already served. The ratio is therefore at most two to one.

Modified O algorithm: Server **A** serves the customer in queue **A** with shortest time left. Server **O** serves the customer with shortest time left. If there is one in both queues then it serves the one in queue **O**.

Theorem 3.3: For the case where each customer can have an attached time of one or zero the modified **O** algorithm is optimal.

The proof follows immediately from the following observations:

1) It is always optimal to serve a customer with an attached time of zero.

2) Between two customers with equal attached time is is always better to serve the one on queue **A**.

Theorem 3.4: For the case where the customers can have an attached time greater than one the competitive factor of the modified **O** algorithm is two.

Proof of upper bound: The same as for theorem 2 with the customers arriving at time one and two instead of zero and one.

Proof of lower bound: The same as for theorem two.

Theorem 3.5: For the model as above there is no non-deterministic algorithm with a competitive factor better than 4/3.

Proof: The infinite repeating sequence that yielded a competitive factor of two for the modified **O** algorithm, with the modification that it is randomly decided with equal probability whether the second server will be of type **A** in turn one or of type **O** in turn two will yield a competitive factor of at least 4/3 for any non-deterministic algorithm.

4. A two queue three server model

In this model there are two queues **A** and **B**. There are three types of servers: **O** that can serve in any of the queues, **A** and **B** that can serve only in the appropriate queue. Each customer has an attached value. In this scenario we will assume that those values are equal. Each new turn three new customers appear divided between the two queues (in any way) and three servers **A**,**B** and **O**. These customers have no attached time (i.e., attached time equals infinity). The objective function is to maximize the average number of servers used per turn over infinite sequence (value).

Some observations about the model:

1) Regardless of the algorithm and the sequence, the total number of customers in the queues is non decreasing.

2) In every turn and for every sequence every algorithm will be able to serve at least two customers.

Prefer A algorithm: **O** will serve **A** if there is a second customer in it, else it will serve **B**.

Theorem 4.1: The competitive factor of the Prefer **A** algorithm is 6/5.

Proof of upper bound: Figure 2 demonstrates an infinite sequence that has a competitive factor of 6/5.

The infinite repeating sequence shown in figure 2 - two in **A**, one in **B**, followed by three in **B**, with the initial stage zero in **A** and two in **B** - has a competitive factor 6/5. The optimal algorithm will always be serving one in **A** and two in **B** while the Prefer **A** algorithm will be serving two in **A** and one in **B** followed by two in **B** and none in **A**.

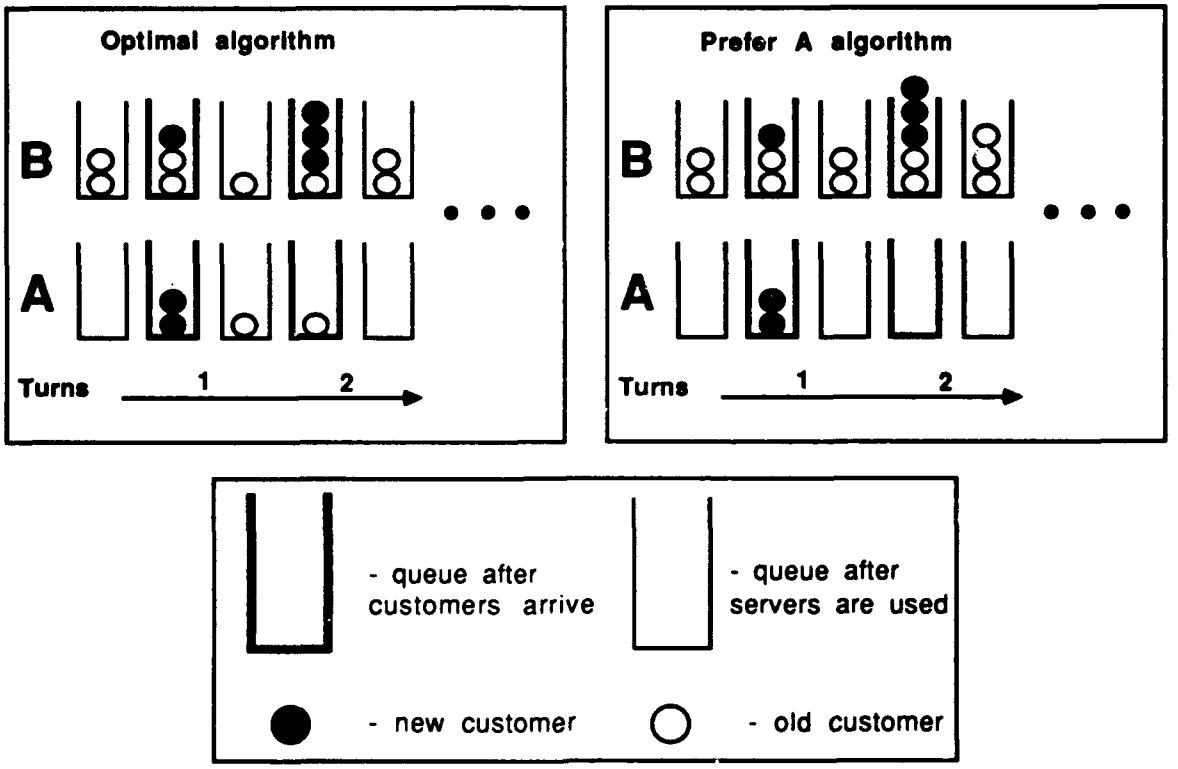


Figure 2. Repeating sequence with a competitive factor of 6/5

Proof of lower bound:

Lemma 4.1: When compared to any other algorithm on any sequence the number of customers in queue **A** will be less than or equal to that of the other algorithms while the number in queue **B** will be larger or equal.

Proof: Trivial.

Lemma 4.2: At any given turn Prefer **A** can serve at most one more from queue **A** than any other algorithm..

Proof: If Prefer **A** can serve two from queue **A** while the other algorithm cannot serve even one then it is in contradiction to Lemma 4.1.

Lemma 4.3: Whenever Prefer **A** serves more from queue **A** it serves three customers.

Proof: Since the other algorithm did not take two from queue **A** (and it had two to take by Lemma 4.1) it must have taken two from **B**. By Lemma 4.1 if any algorithm had customers in queue **B** so did Prefer **A** and therefore Prefer **A** served three customers in that turn.

Lemma 4.4: The only possible case in which Prefer **A** will serve less than any algorithm is when Prefer **A** serves two from **B** and the other algorithm serves three. In that case the second algorithm serves at least one from **A**.

Proof: The only other possible case is when prefer **A** serves two from queue **A** and has none to serve from queue **B**. From Lemma 4.1 we see that in that case any other algorithm will have none to serve from queue **B** as well.

In order for the case of Prefer **A** to have nothing to serve from queue **A** while the optimal algorithm does, it must have served more from queue **A** than the optimal algorithm before. Since the number of times it serves only two from **B** while the optimal served three is less than or equal to the number of times it served more from queue **A**, then for every time it served two there is another time it served three and therefore the ratio is 6/5.

I assumed that the optimal algorithm served three customers each turn. Since we already proved the upper bound this is permissible.

Theorem 4.2: There is no algorithm with a worse competitive factor than Prefer **A**.

Proof: Let **Q** be a worse algorithm. The number of unserved customers must be increasing. If it is increasing in more than one queue then **Q** will be able to serve three customers in each turn. It is therefore increasing in only one queue. Without loss of generality let this queue be queue **B**. We know that **Q** cannot cause queue **B** to grow faster then Prefer **A** - contradiction.

Definition: The Greedy algorithm is one in which **O** serves the second customer with the largest value.

Theorem 4.3: The greedy algorithm is identical to Prefer **A** for the case in which the customers in queue **A** have a larger value.

Proof: Trivial.

The greedy algorithm is not optimal under the objective function of minimal number of kidney machines as we will need a machine for every customer in the queue.

Definition: The Long algorithm is one in which **O** will serve the longer queue.

Theorem 4.4: The Long algorithm has a competitive factor of one.

Proof: In order for the competitive factor to be larger than one, one of the queues has to grow without a bound. If there is a bound on the growth of the queues then the number of times the algorithm served fewer customers than the optimal algorithm is bounded by a constant and the competitive factor is one. Since one of the queues is supposed to grow and from that queue the long algorithms will take two, it will be able to take from the other queue as little as possible. It will serve two only when the queue with fewer customers is empty, since Long algorithm took as few as possible from it this queue will also be empty for the optimal.

Definition: The Rand algorithm is one in which server **O** will chose randomly with equal probability between queues **A** and **B** (if given the choice).

Theorem 4.5: The Rand algorithm has a competitive factor of 9/8.

Proof: We know that in order for the competitive factor to be greater than one, one of the queues has to grow without a bound. Half of the time when the Rand algorithm will make a choice it will make the "good" choice of choosing from the longer queue. As we saw in the analysis of the Prefer **A** algorithm you can "pay" at most once for every bad choice. Therefore you "pay" in average at most once for every to choice. When you have a choice you serve three, when you pay you serve two. The competitive ratio is therefore 9/8.

An example of this happening is when the serving sequence is two to **A** and one to **B** for **N** turns, then one to **A** and two to **B** for **N** turns, and then three to **B** for **N** turns. After 2***N** turns on average queue **A** will be emptied by the Rand algorithm and for the next **N** turns the Rand algorithm will be able to serve only 2***N** customers. The probability of queue **A** not emptying after (2***N**+any fixed fraction of **N**) number of turns approaches zero as **N** approaches infinity.

Definition: The Modified Rand algorithm is one in which server **O** will choose randomly a patient from one of the queues with equal probability to choose any patient.

Theorem 4.6: The Modified Rand algorithm has a competitive factor of 1.

Proof: The proof of this case is similar to the proof for the Long algorithm. The probability of choosing from the more occupied queue approaches one and therefore the rate of wrong moves approaching zero. Since the cost incurred serving in a different way than the optimal algorithm is at most a constant, the amortized performance approaches that of the optimal algorithm.

For this model, which is a gross simplification of the general model, we see that in order to have a good allocation policy we must look not only at the person who is first in line but also at the length of the line. We saw that algorithms that take the length of the line into account (the Modified Rand algorithm can be conceived as such an algorithm) can get a competitive factor of one while others are not quite as efficient. Since real problems can be constructed which are equivalent to this model we conclude that in the general model the ability to look at the distribution is important.

5. A queue model with balking customers

In this model there are two queues - **A** and **B** - with a single server attached to each. There is another server - **O** - that on each turn can help in one of the queues. Each server can serve one customer per turn. All customers have the same value. A customer has an attached time (number of turns) within which it must be served. This time is zero or one turns. Each new turn three new customers appear divided between the two queues (in any way). The objective function is to maximize the average number of organs used per turn over an infinite sequence.

Notation: The matrix (**x**,**y**,**z**,**w**) denotes **x** customers in **A** which will balk from the queue if not served immediately, **y** customers in **B** which will balk from the queue if not served immediately, **z** customers in **A** willing to wait one turn, and **w** customers in **B** willing to wait one turn.

Observations about the model:

1) This is a finite model; the number of customers in both queues combined after each turn is at most three, all of which will be old customers by the beginning of the next turn (old in the sense that they will not be willing to wait any longer).

2) In every turn and for every sequence every algorithm will be able to serve at least two customers (notice that the number of customers might be decreasing).

3) In every reasonable algorithm an old customer will always be preferred over a new one.

4) There is no algorithm with a competitive factor of one.

5) In the case of a choice between two customers in a queue and one customer in the other it is never advantageous to serve the single customer.

6) Because of observation 5, the only choice a best algorithm can make is when after the A and B servers served the matrix is (0,0,1,1).

7) In that case a random algorithm will have a better performance against an oblivious adversary as at least half the time it will choose the right choice.

Theorem 5.1: For this problem the competitive case of the best deterministic algorithm is 6/5.

Assume without loss of generality that in the symmetric case of (0,0,1,1) the best algorithm will choose from **A**. Because of symmetrical considerations it doesn't matter which way it chooses.

Proof of upper bound: The starting position is (1,0,0,0), the cycle is (0,1,1,1) (2,0,1,0). In each cycle our algorithm will serve five while the optimal will serve six.

Proof of Lower bound: In order for the optimal algorithm to perform better there has to be a difference between the ways that the servers are utilized. The only difference is when there is a choice for server **O**, in that case the algorithm serves 3. In a worst case scenario in the turn after the different choice the optimal will serve more, i.e. the optimal serves three while the best deterministic serves two. Therefore the lower bound is 6/5.

Corollary: Every deterministic algorithm in this model has a competitive factor of 6/5.

This result implies that we are close to the end of the usefulness of using the online model to analyze the organ allocation policy. It is clear that some algorithms are better (or at least as good) than others (on every sequence) but the competitive factor is still the same. This counter intuitive result stems from the power of the adversary. If an algorithm has a "weakness" the adversary will give it only sequences that exploit this weakness. In this model it makes no sense to talk about expected future distribution of customers and servers. It is therefore clear that the problem needs to be analyzed in a different domain to show that these criteria are important.

Theorem 5.2: Against an oblivious adversary the competitive factor of the best non-deterministic algorithm is at least 16/15.

Proof: Consider the following repeating three turn sequence. Start with (0,0,3,0) followed by (0,1,1,1) on the second turn. The third turn will be either (3,0,0,0) or (0,3,0,0) with equal probability. We can divide every turn to three parts: beginning of turn, after customers and servers arrive, after servers are used. The beginning of turn **i** is equal to the end of turn **i**-1. For our sequence the situation in the beginning of turn one is always (regardless of the algorithm used) (0,0,0,0). The situation after customers and servers arrive in turn two is always (1,1,1,1). Any algorithm will be able to serve three customers in turn three only if it served the right customers in turn two. The probability of serving the right customers in turn two is at most 1/2. Therefore no non-deterministic algorithm will serve on average more than 7.5 customers per three turns (2 on the first ,3 on the second, and 2.5 on the third) while the optimal algorithm will serve 8. It is clear that no non-deterministic algorithm will have a competitive factor better then 16/15.

Figure 3 demonstrates a possible three turn sequence for the optimal and a non-deterministic algorithm. In the figure we show only the second and third stages of each turn. As the last stage of one turn and the first stage of the following are equal we are not missing any information.

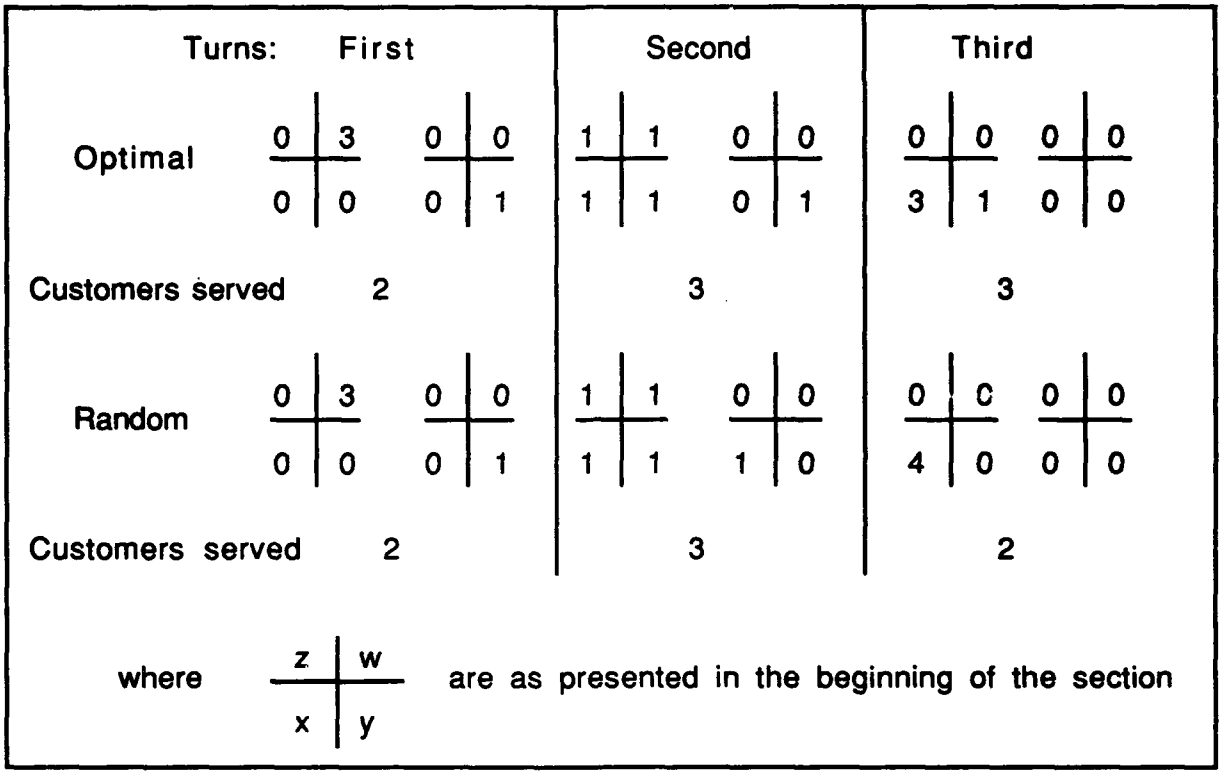


Figure 3. Comparison of two algorithms on a possible three-turn sequence

6. Competitive factor of a finite state deterministic algorithm

Let **R** be a finite state deterministic algorithm (for our example, but without loss of generality, one that is used for the model in section 5). We show how to calculate its competitive factor.

Definition: A cost automata is an automata with a cost associated with each edge.

Consider the following cost automata for the deterministic algorithm **R**:

States are the possible positions in the beginning of a turn .

From each state in the automata 20 edges (corresponding to the possible ways for customers to arrive and labeled accordingly) connect to the other states. Associated with each edge is a cost function which represents the number of customers that were served by traversing that edge.

The reader should convince himself at this point that if two algorithms have the same cost automata then those two algorithms are identical.

Definition: A circuit is a collection of edges {(i,j)...(k,l)} such that it can be ordered

with the terminal of edge **i** being the same as the source of edge **i+1** and the terminal of the last edge being equal to the source of the first edge. No subset of a circuit should have this property.

Lemma 6.1: The competitive factor of **R** is equal to the competitive factor of the circuits with the worst competitive factor in the automata.

We assume that given an infinite repeating sequence the optimal algorithm can be computed as the future is known. It is therefore possible to compute the competitive factor of **R** on a given starting state and an infinite repeating sequence, which we will call the competitive factor of the circuit.

Proof: Assume by way of contradiction that **R** has a worse competitive factor than any circuit (if there is a circuit with a worse competitive factor then the sequence represented by repeating the circuit has a worse competitive factor - contradiction). Let **Q** be the sequence on which the algorithm performs worst when compared to the optimal algorithm. Since **Q** is an infinite sequence(we need only longer then the number of states in the automata), and since the automata is finite, **Q** must contain circuits. Let's look at one of those circuits. If the competitive factor of that circuit is equal to **Q** we are done. Else remove the circuit from **Q** and the result **Q'** will have a worse competitive factor than **Q** - contradiction to **Q** being the worst.

Lemma 6.2: There are eight reasonable algorithms for the model represented in 5.

Proof: From observation 3 in section 5 we see that the only possible cases in which different algorithms will behave differently is when server **O** can serve customers which can wait two turns in queue **A** and **B**. There are three such cases, therefore there are eight possible algorithms (three of which are mirror images of the other three).

Theorem 6.3: It is possible to determine the best competitive factor of a deterministic algorithm by using exhaustive search.

Proof: As the number of algorithms is finite and the graph is finite one can find all possible circuits for each algorithm and check them all (although it is exponential).

7. Summary

This paper had two main thrusts. One was to show that organ allocation policy should consider the current distribution of patients as well as the expected future distribution of patients and organs. We showed that current distribution of patients should be considered by the allocation algorithm. We showed (section 4) that an algorithm that considers the distribution of patients is better than algorithm that doesn't - it might even be optimal. Because in competitive analysis future distribution has no meaning, since only worse case scenarios are considered, we didn't show that expected future distribution should play a role in the organ allocation policy. Work on finding an optimal algorithm under a more realistic model is in progress. The models that are being investigated are the general queue model with random arrival of customers and servers. We have already shown, using statistical analysis on some special cases, that the optimal policy depends on the current distribution of patients as well as the expected future distribution of patients and organs.

The second major thrust of the paper was competitive analysis of queue models including those in which customers may balk. We presented the general model, and showed under what conditions we can expect a competitive algorithm to exist. We explored the differences between deterministic and non-deterministic competitive algorithms for different subsets of the general model. We introduced the concept of cost automata and explained how it can be used to calculate the competitive factor of any finite state deterministic algorithm. We also showed how to calculate the competitive factor of algorithms in some cases for which the number of states of the problem is infinite.

Acknowledgments.

I would like to thank Daniel Sleator for introducing me to online analysis, and Michael Trick for his active involvement from beginning to end, his valuable criticisms, and especially for asking the right questions.

References

[1] Tim Breen , private comunications.

[2] Shai Brill, "Donor Recipient HLA-A,B Matching Probability for Cadaver Kidney Transplantation in the Israeli Population" , PhD proposal in Tel-Aviv University.

[3] I. David, "One-Attribute Sequential assignment Match Processes in Discrete Time." To be Published in Operations Research.

[4] Harrison, Principles of internal madicine, vol 2, 12th ed. pp 1160,1163

[5] R. Jean Ruth. Leon Wyszewianski and Gary Herline, "Kidney Transplantation: A Simulation Model for Examining Demand and Supply." Management Science Vol. 31, No 5, May 1985.

[6] H. Tristram Engelhardt, Allocation of Scarce Medical Resources and the Availability of Organ Transplantation: Some Moral Presupposition, Bioethics, New Jersey, Paulist Press.

[7] UNOS Article of Incorporation as of October 3, 1990.

[8] UNOS "It's a Miracle" a flyer

[9] Wayne L. Winston, Operation Research Application and Algorithms,Boston, PWS-Kent Publishing Company